R Packages for health economic evaluation: A tutorial.

Robert Smith, Wael Mohammed & Paul Schneider

R for HTA annual workshop
University of York
9th June 2023

DARK PEAK ANALYTICS

University of Sheffield

Abstract

Background: The use of programming languages such as R in health economics and decision science is increasing, and brings numerous benefits including increasing model development efficiency, improving transparency, and reducing human error. However, there is limited guidance on how to best develop models using R. So far, no clear consensus has emerged.

Methods: We present the advantages of creating health economic models as R packages - structured collections of functions, data sets, tests, and documentation. Assuming an intermediate understanding of R, we provide a tutorial to demonstrate how to construct a basic R package for health economic evaluation. All source code used in or referenced by this paper is available under an open source licence.

Results: We use the Sick Sicker Model as a case study applying the steps from the tutorial to standardise model development, documentation and aid review. This can improve the distribution of code, thereby streamlining model development, and improve methods in health economic evaluation.

Conclusion: R Packages offer a valuable framework for enhancing the quality and transparency of health economic evaluation models. Embracing better, more standardised software development practices, while fostering a collaborative culture, has the potential to significantly improve the quality of health economic models, and, ultimately, support better decision making in healthcare.

Who we are

Dr. Paul Schneider

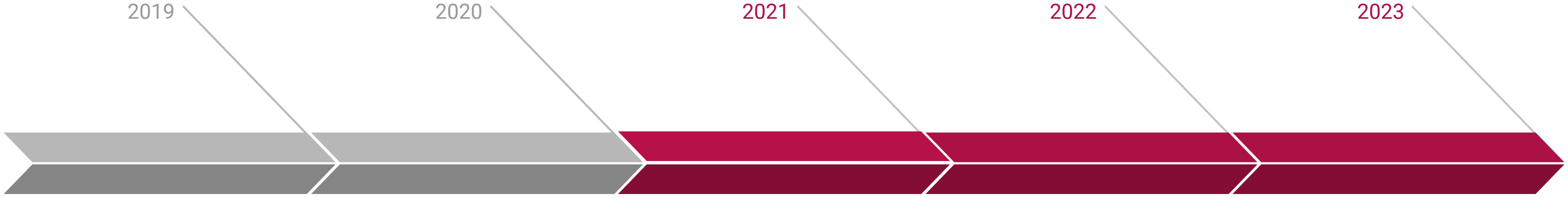
Dr. Robert Smith

Dr. Sarah Bates

ShangShang Gu

Wael Mohammed

R-HTA timeline



2019

Attended

2020

Attended

2021

Making Health Economic Models Shiny: A tutorial

Smith RA and **Schneider PP.** Making health economic models Shiny: A tutorial. *Wellcome Open Res* 2020, **5**:69 (https://doi.org/10.12688/wellcomeopenres.15807.2)

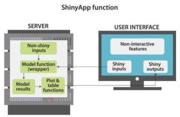
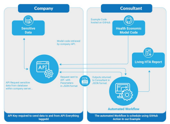


Figure 1. Diagram depicting how the Sick-Sicker app is structured.

2022

Living HTA: Automating Health Economic Evaluation with R

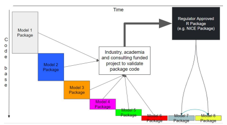
Smith RA, Schneider PP and **Mohammed W**. Living HTA: Automating Health Economic Evaluation with R. *Wellcome Open Res* 2022, **7**:194 (https://doi.org/10.12688/wellcomeopenres.17933.2)


Company
Consultant
Sensitive Data
Health Economic Model Code
API
Living HTA Report
Automated Workflow
API Key required to send data to and from API Everything logged
The automated Workflow is schedule using GitHub Action in our Example

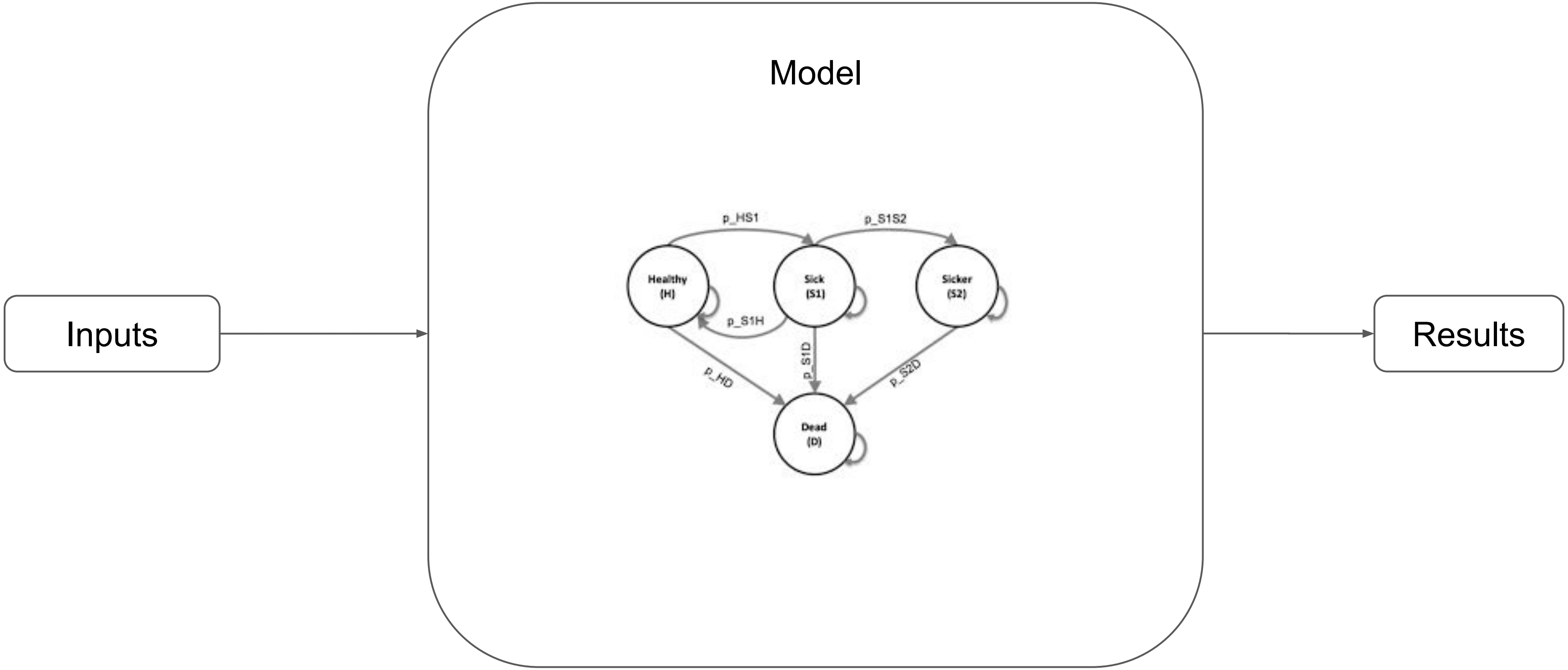
2023

R Packages for health economic evaluation: A tutorial

Smith RA, Mohammed W and **Schneider PP**. R Packages for health economic evaluation: A tutorial. 2023. [Draft paper currently under review in GoogleDoc]()



Building a model in R

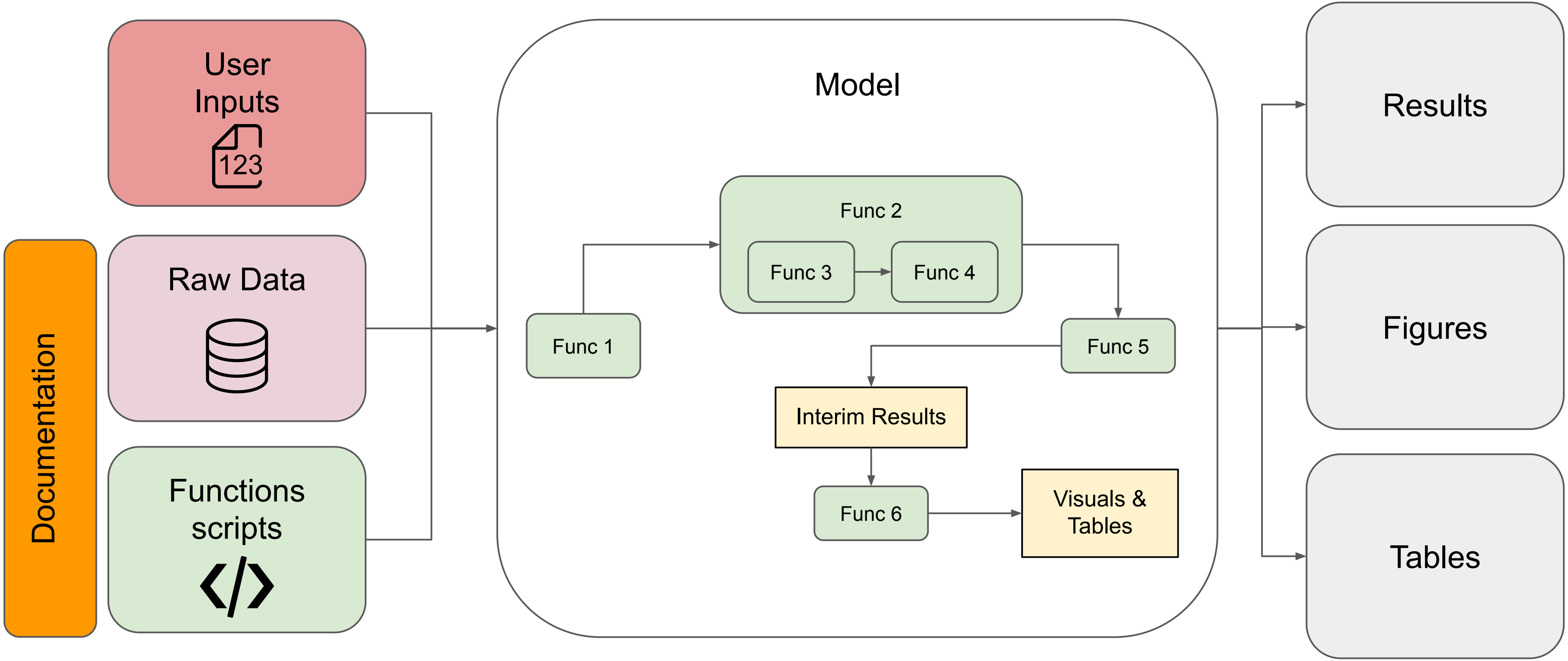


Inputs

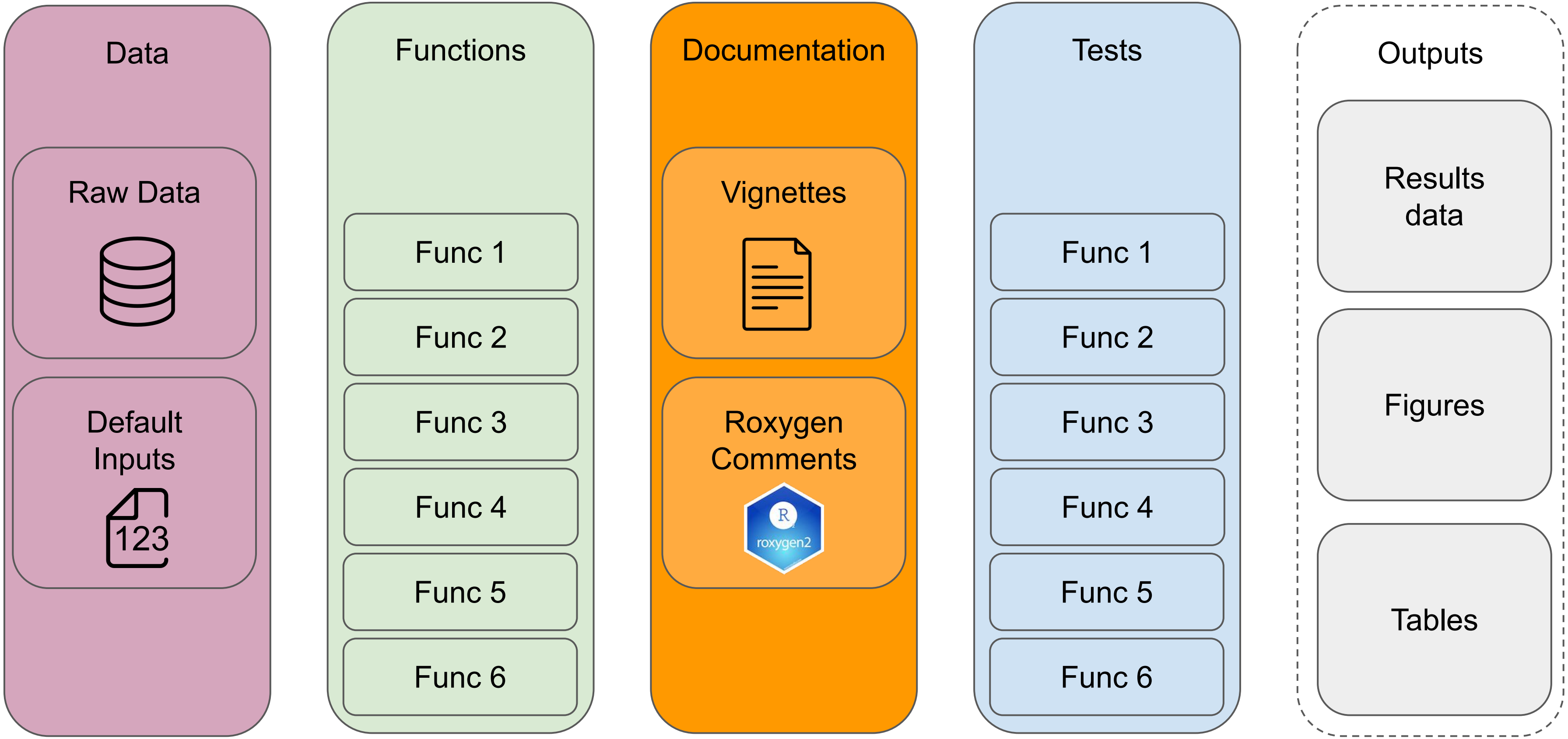
Model

Results

Building a model in R



Building a model in R



Data

- Raw Data
- Default Inputs

Functions

- Func 1
- Func 2
- Func 3
- Func 4
- Func 5
- Func 6

Documentation

- Vignettes
- Roxygen Comments

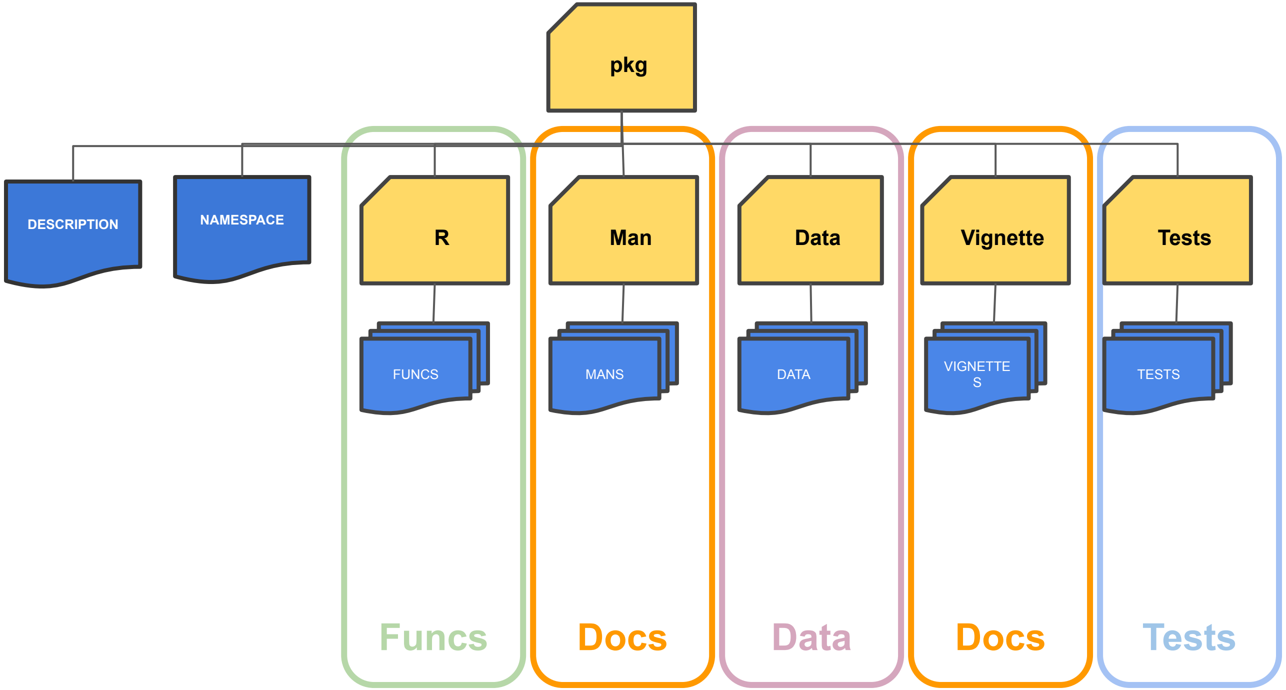
Tests

- Func 1
- Func 2
- Func 3
- Func 4
- Func 5
- Func 6

Outputs

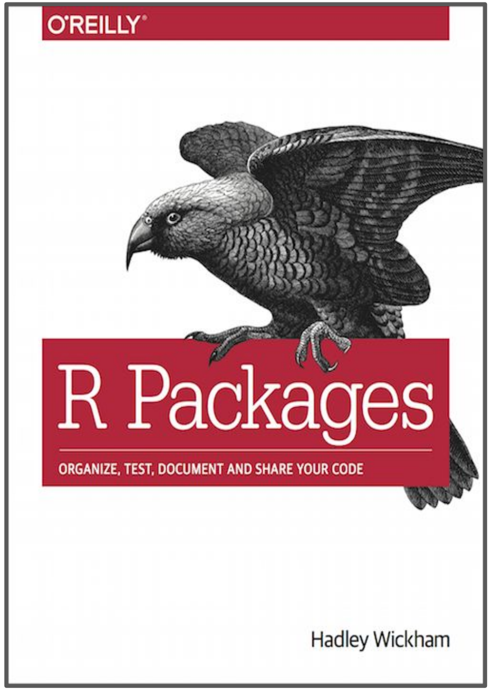
- Results data
- Figures
- Tables

Building a package in R



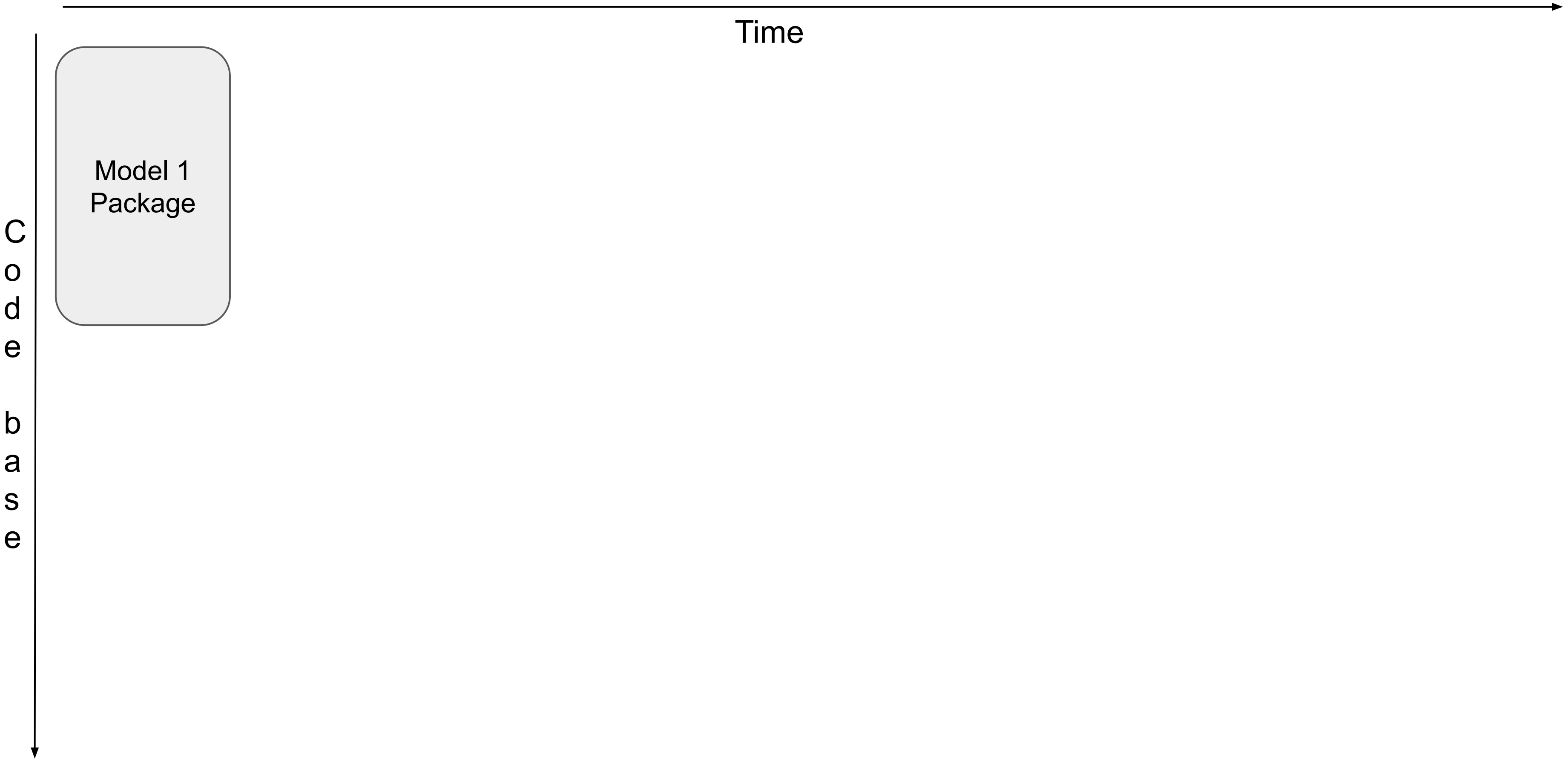
Benefits of using a Package vs non-packaged code.

1. Every Package will have a similar structure
    - a. Improves familiarity with models.
2. Documentation by default
    - a. Vignettes to show how the package works (walking the user through the code).
    - b. Roxygen comments on every function (exactly what is it doing)
3. Unit testing is built-in
    - a. Testing gives modeller confidence in their methods.
    - b. Testing allows reviewers to ‘test the tests’ rather than from scratch.
4. Functions are more easily distributed (e.g. install_github(“your-package”))
    - a. Therefore don’t have to continually re-invent wheels
    - b. Standardisation (pros and cons)
    - c. Validation (easier to review, more confidence)



https://r-pkgs.org/

Benefits of using a Package vs non-packaged code.

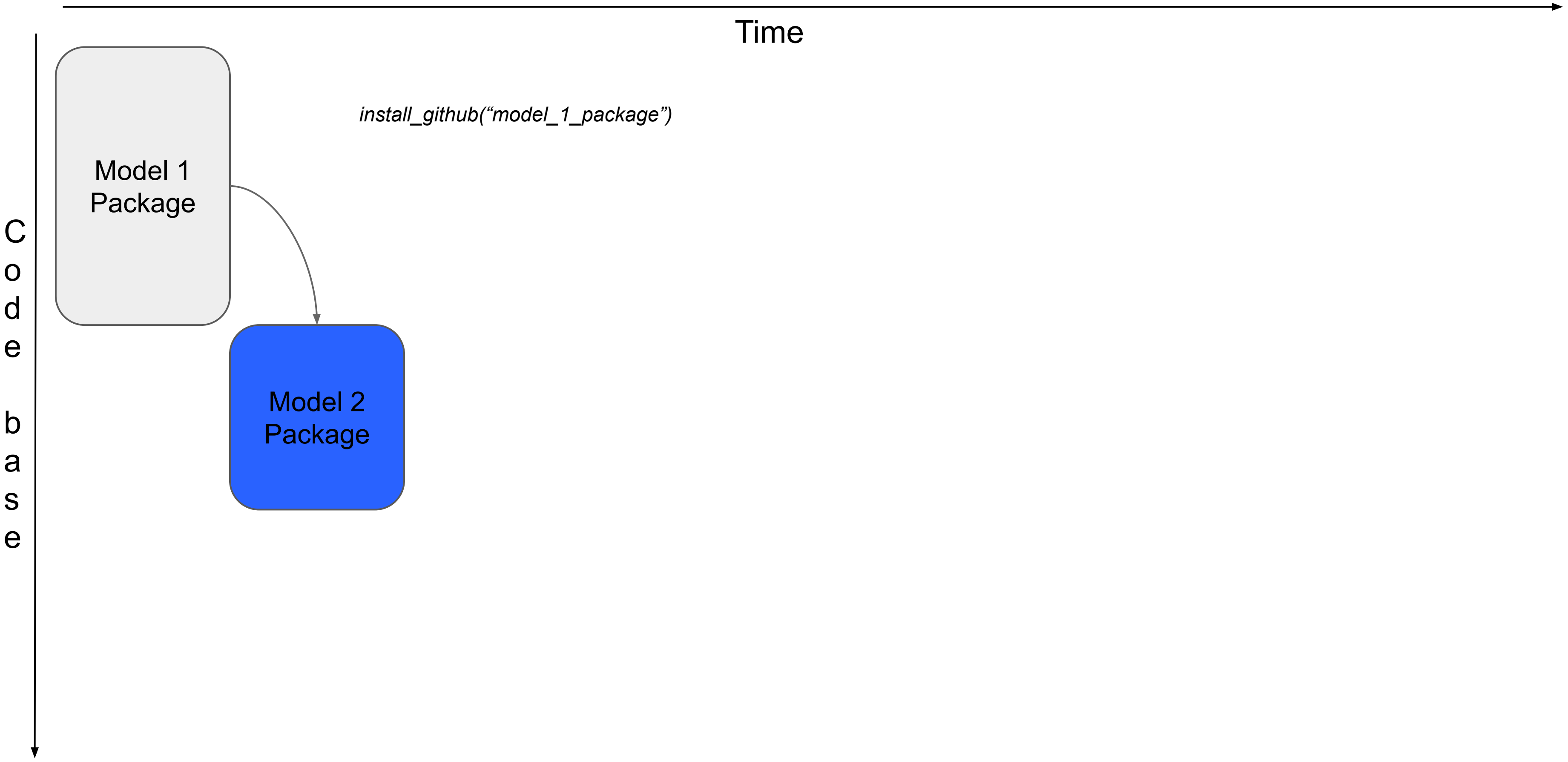


Time

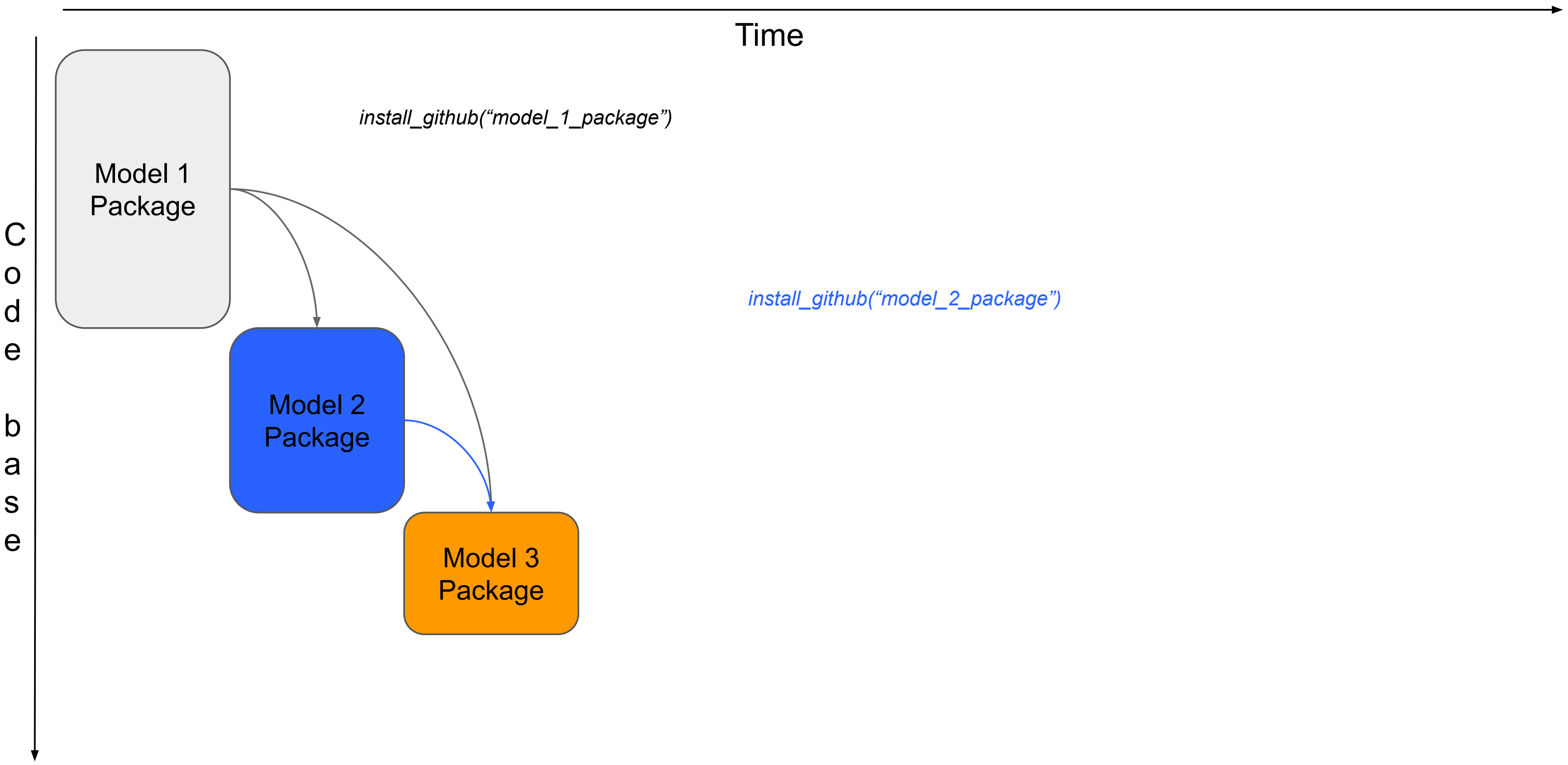
Model 1 Package

Code base

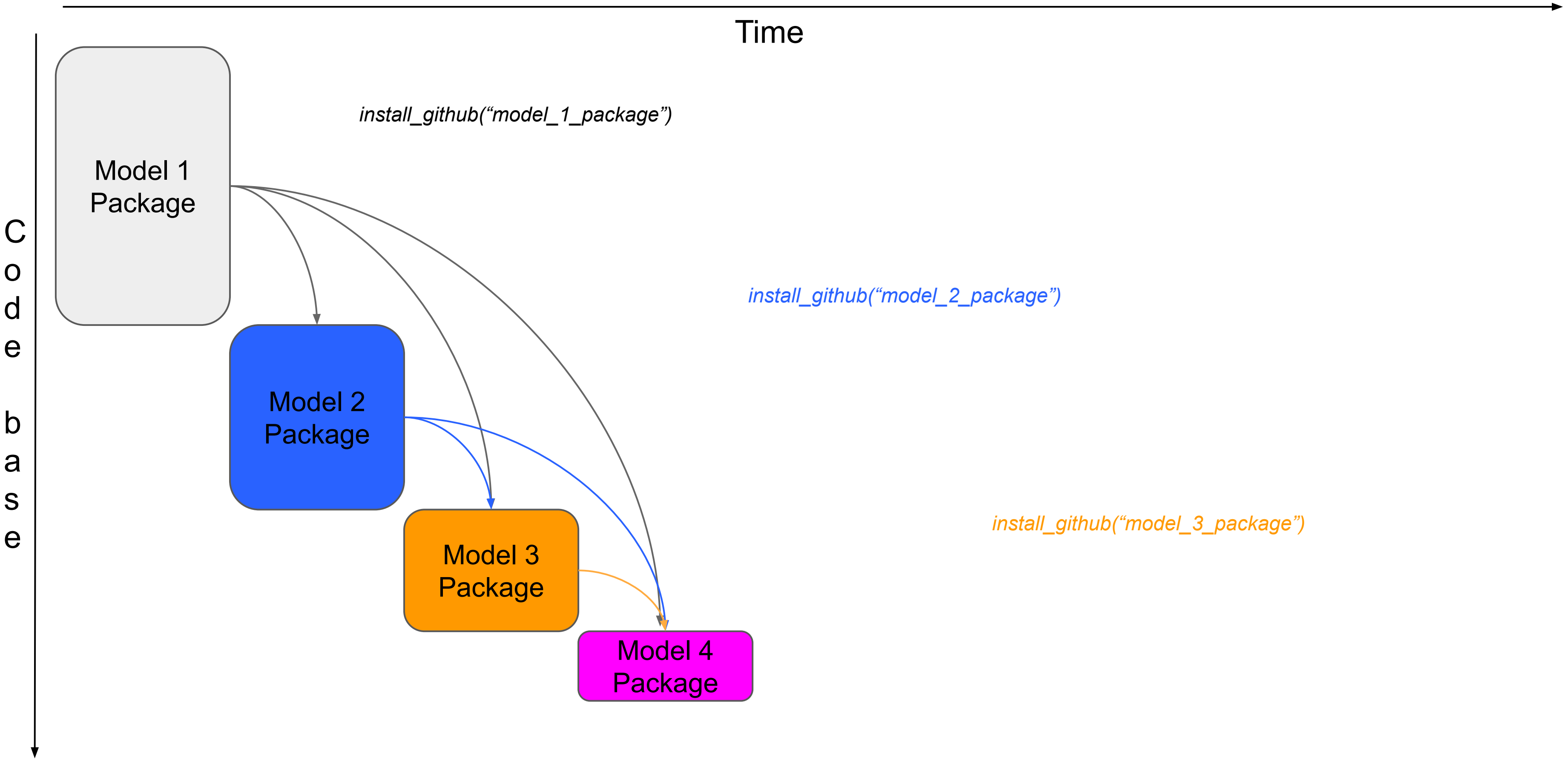
Benefits of using a Package vs non-packaged code.



Benefits of using a Package vs non-packaged code.


Time
Code base
Model 1 Package
Model 2 Package
Model 3 Package
install_github("model_1_package")
install_github("model_2_package")

Benefits of using a Package vs non-packaged code.

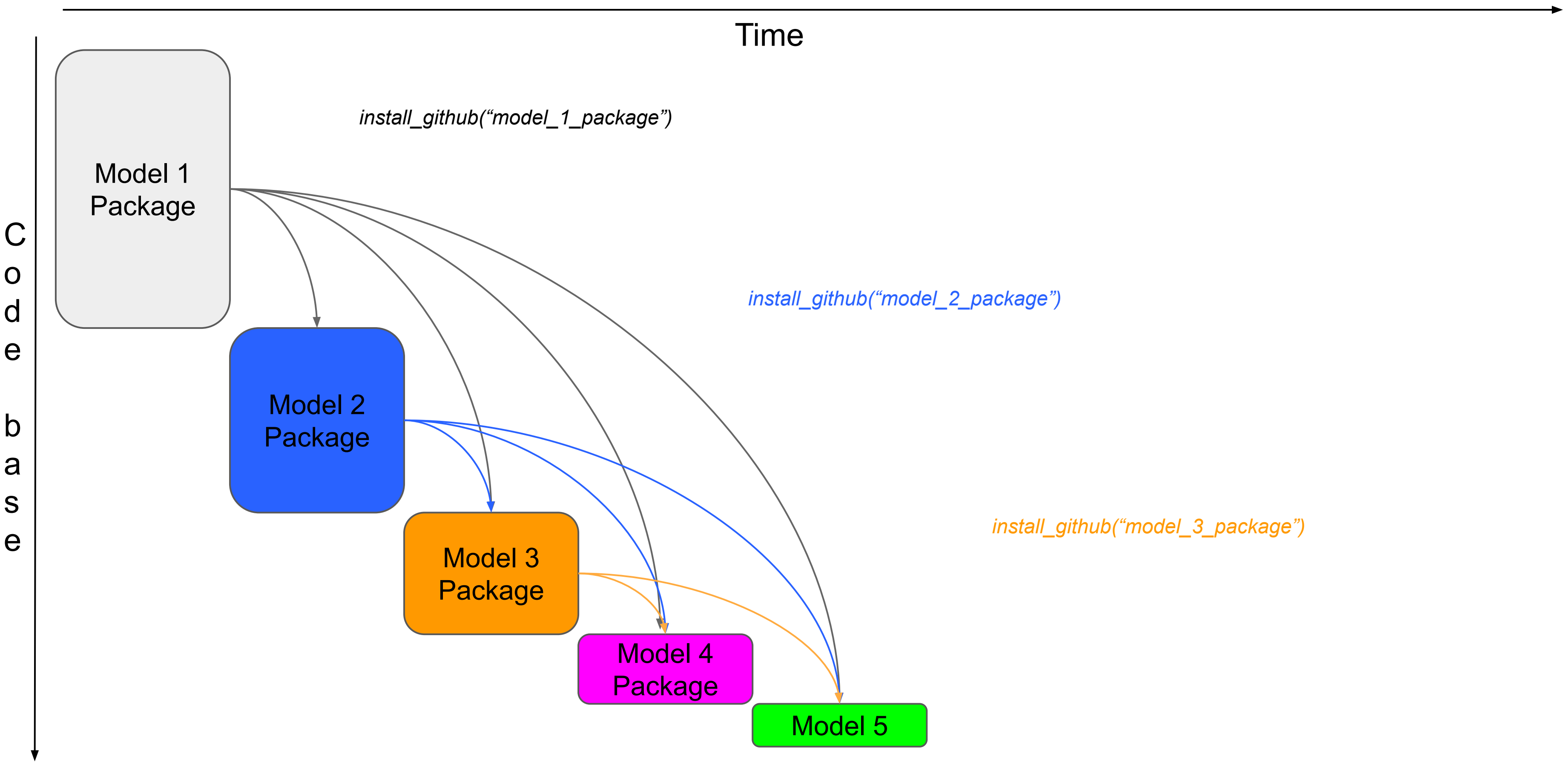


install_github("model_1_package")

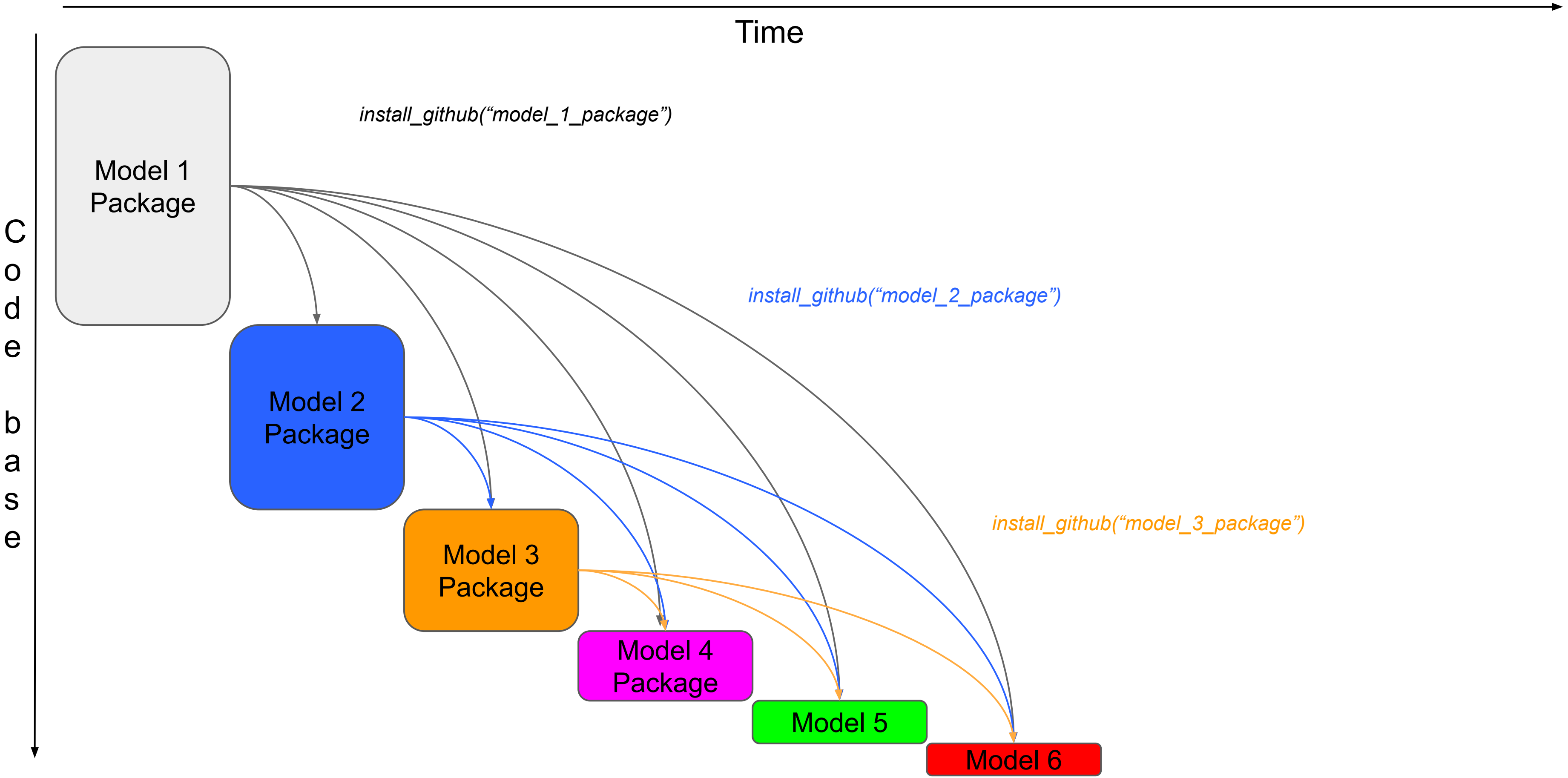
install_github("model_2_package")

install_github("model_3_package")

Benefits of using a Package vs non-packaged code.


Time
Code base
Model 1 Package
Model 2 Package
Model 3 Package
Model 4 Package
Model 5
install_github("model_1_package")
install_github("model_2_package")
install_github("model_3_package")

Benefits of using a Package vs non-packaged code.


Time
Code base
Model 1 Package
install_github("model_1_package")
Model 2 Package
install_github("model_2_package")
Model 3 Package
install_github("model_3_package")
Model 4 Package
Model 5
Model 6

Benefits of using a Package vs non-packaged code.

“But won’t we end up having to install a large number of packages, just to get one or two functions from each?”

Anon

Large package, small function …

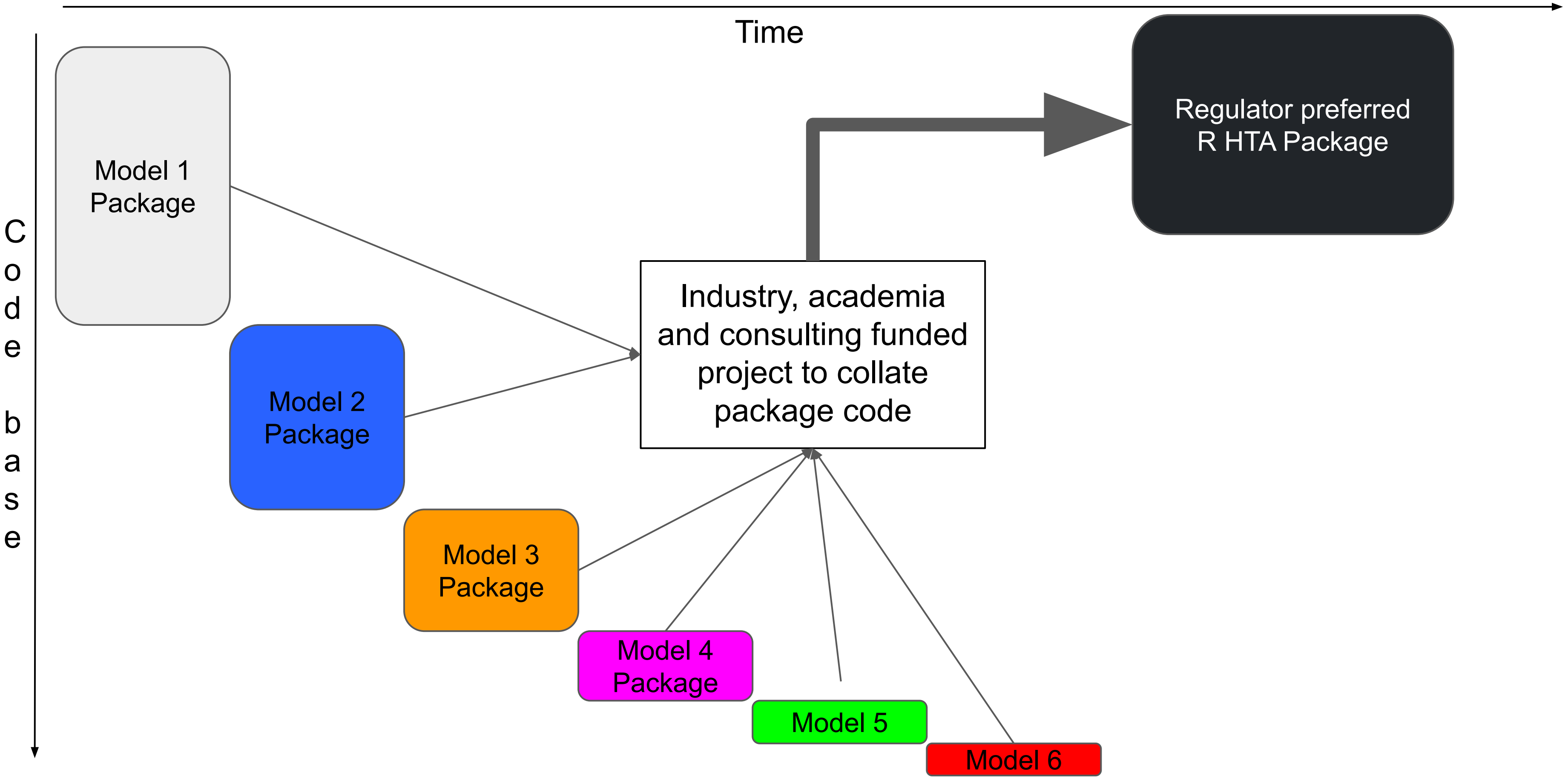
Benefits of using a Package vs non-packaged code.

“But won’t we end up having to install a large number of packages, just to get one or two functions from each?”

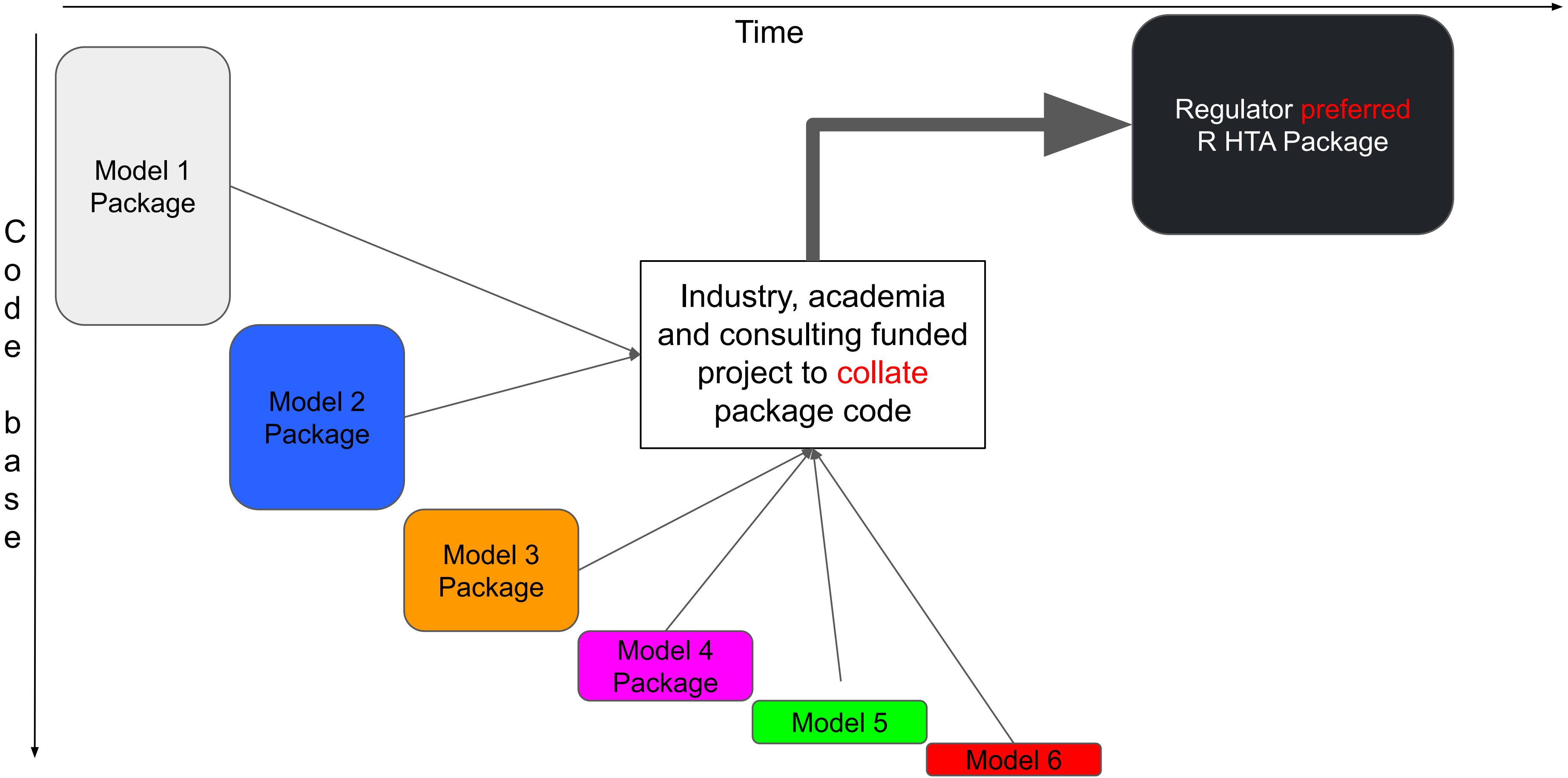
Anon (actually it was Paul)

Large package, small function …

Benefits of using a Package vs non-packaged code.


Time
Code base
Model 1 Package
Model 2 Package
Model 3 Package
Model 4 Package
Model 5
Model 6
Industry, academia and consulting funded project to collate package code
Regulator preferred R HTA Package

Benefits of using a Package vs non-packaged code.



Time

Code base

Model 1 Package

Model 2 Package

Model 3 Package

Model 4 Package

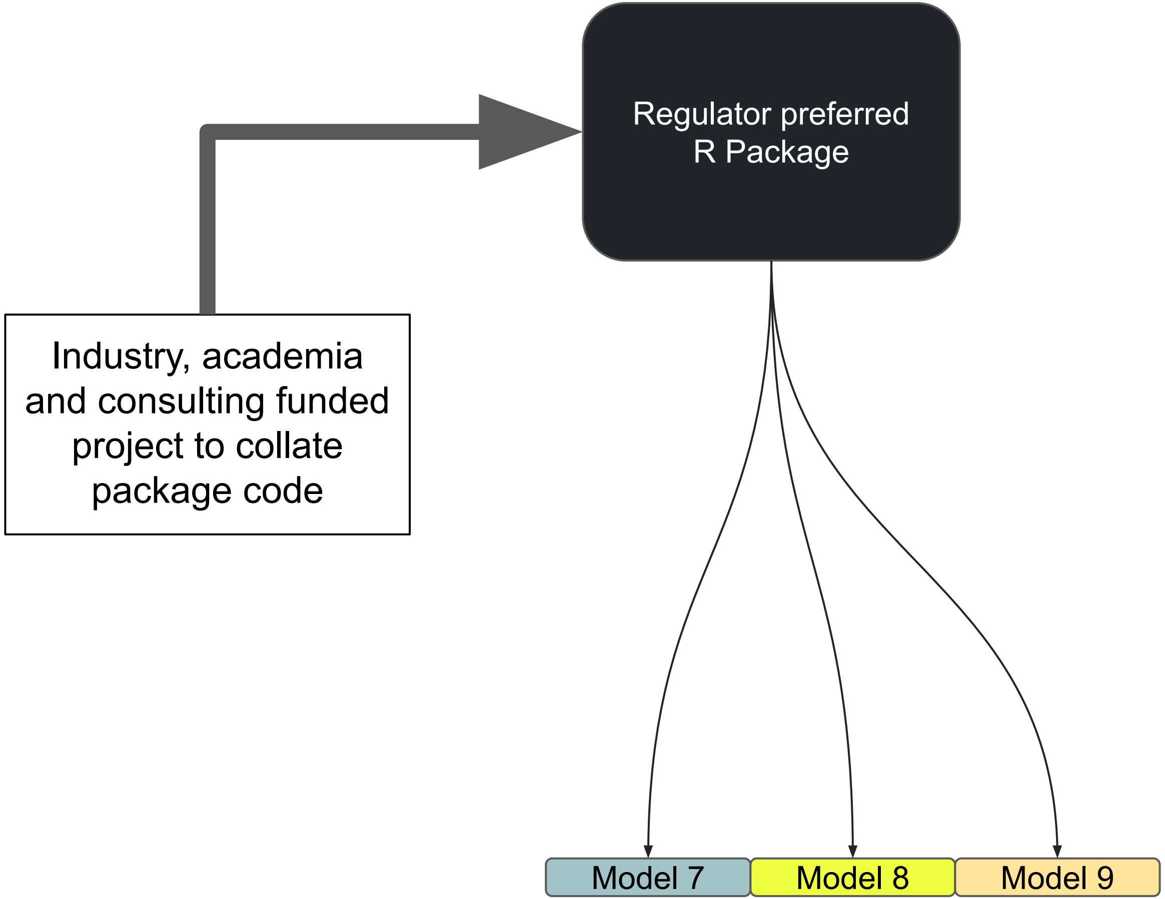
Model 5

Model 6

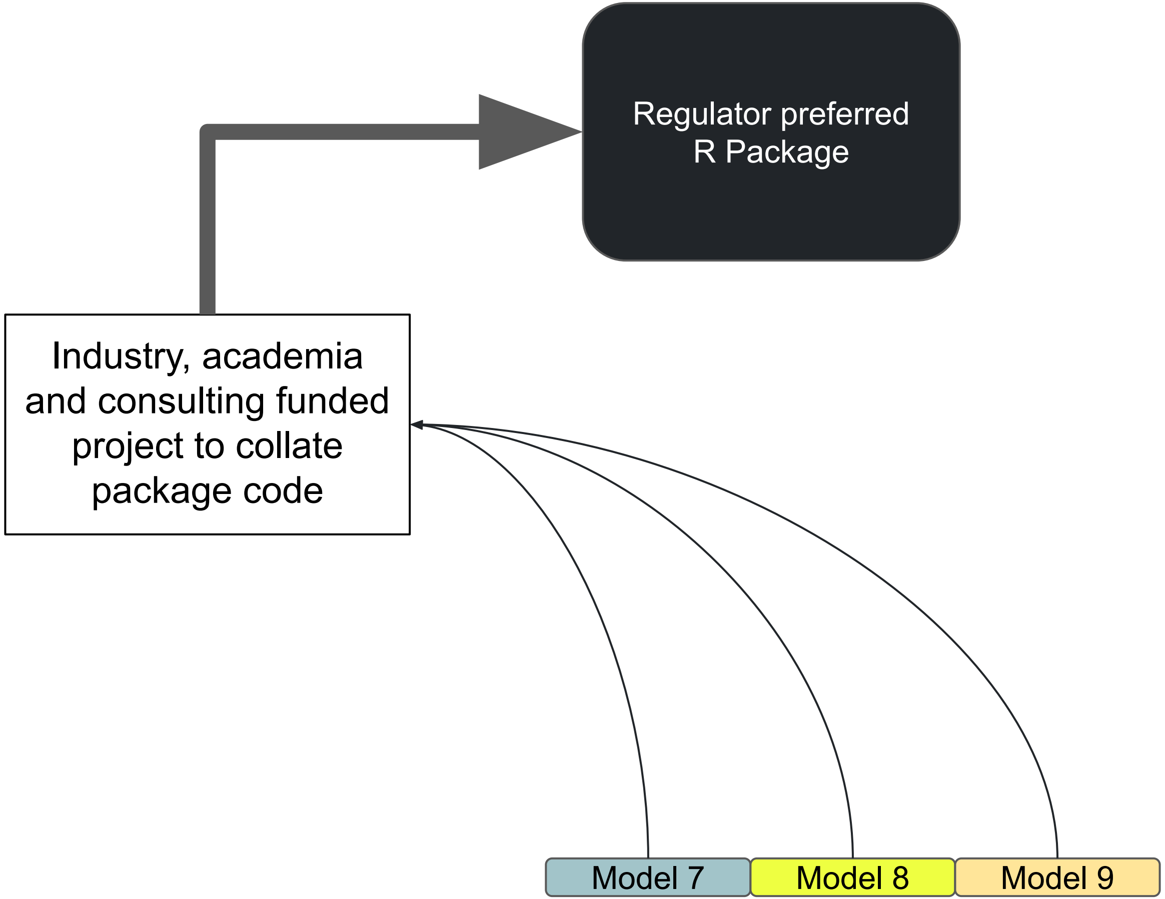
Industry, academia and consulting funded project to collate package code

Regulator preferred R HTA Package

Benefits of using a Package vs non-packaged code.


Regulator preferred R Package
Industry, academia and consulting funded project to collate package code
Model 7
Model 8
Model 9

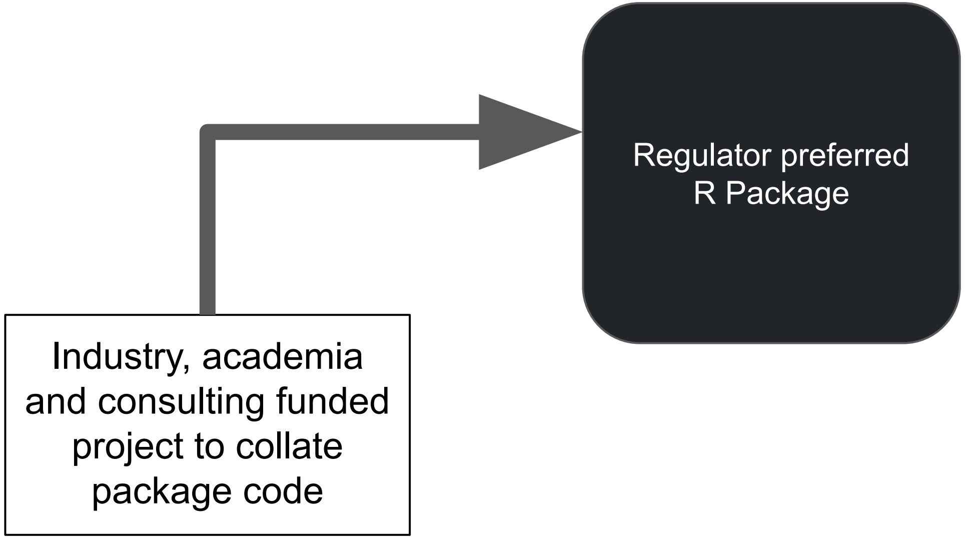
Benefits of using a Package vs non-packaged code.



Regulator preferred R Package

Industry, academia and consulting funded project to collate package code

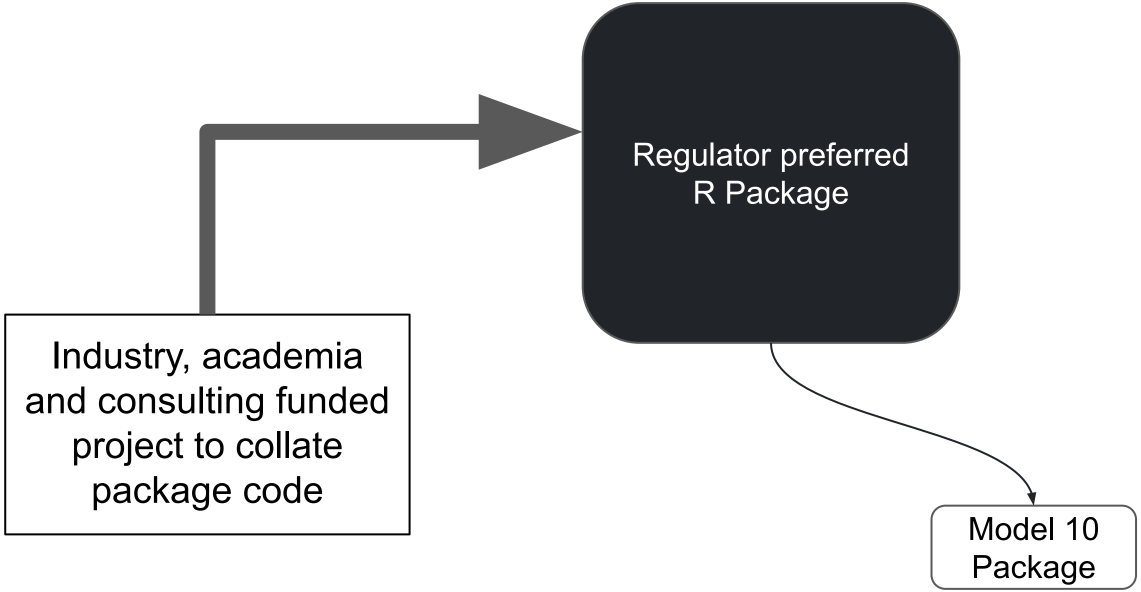
Benefits of using a Package vs non-packaged code.



Industry, academia and consulting funded project to collate package code

Regulator preferred R Package

Benefits of using a Package vs non-packaged code.

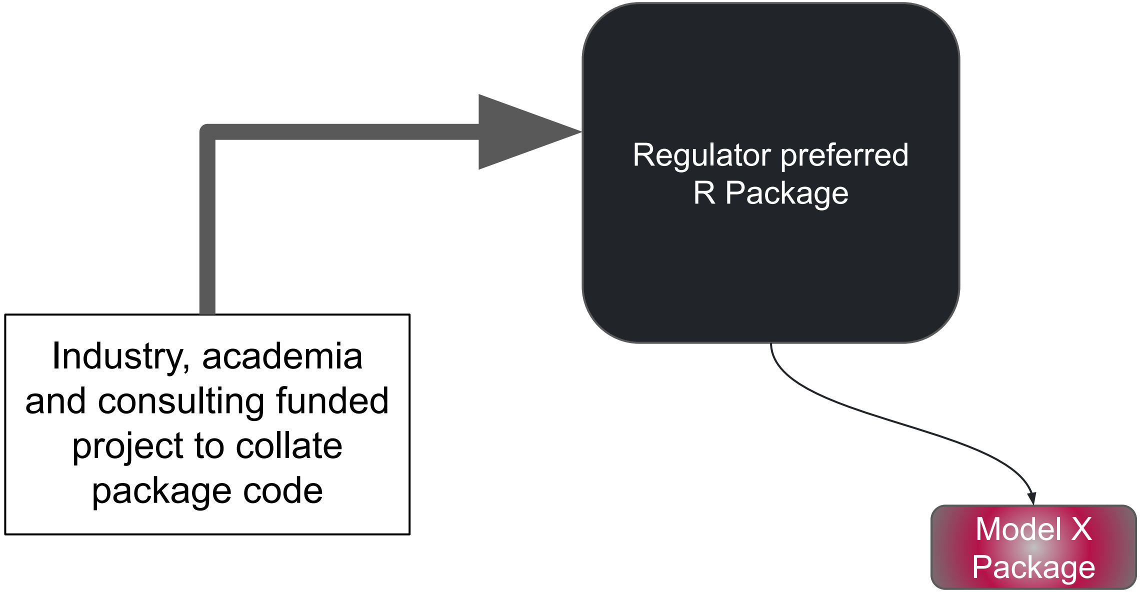


Regulator preferred R Package

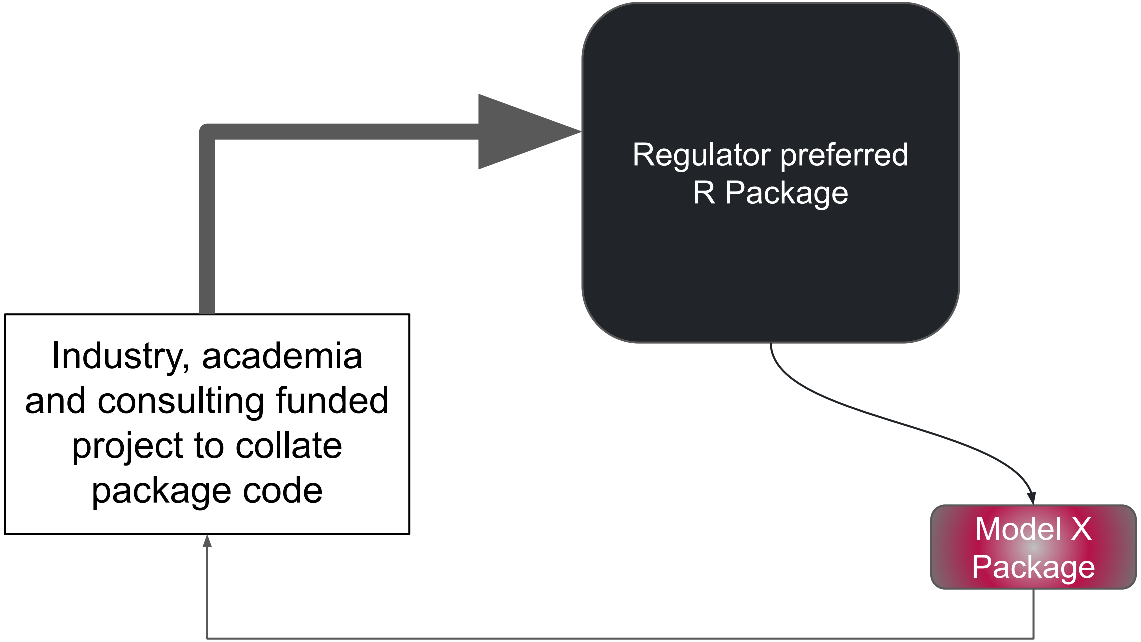
Industry, academia and consulting funded project to collate package code

Model 10 Package

Benefits of using a Package vs non-packaged code.


Regulator preferred R Package
Industry, academia and consulting funded project to collate package code
Model X Package

Benefits of using a Package vs non-packaged code.

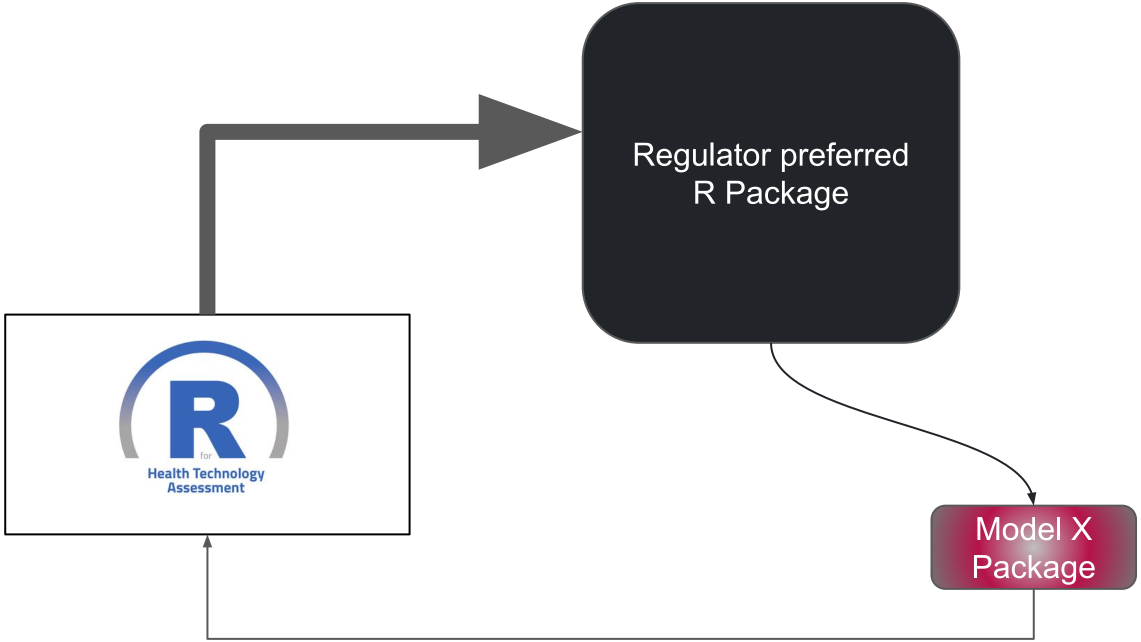


Regulator preferred R Package

Industry, academia and consulting funded project to collate package code

Model X Package

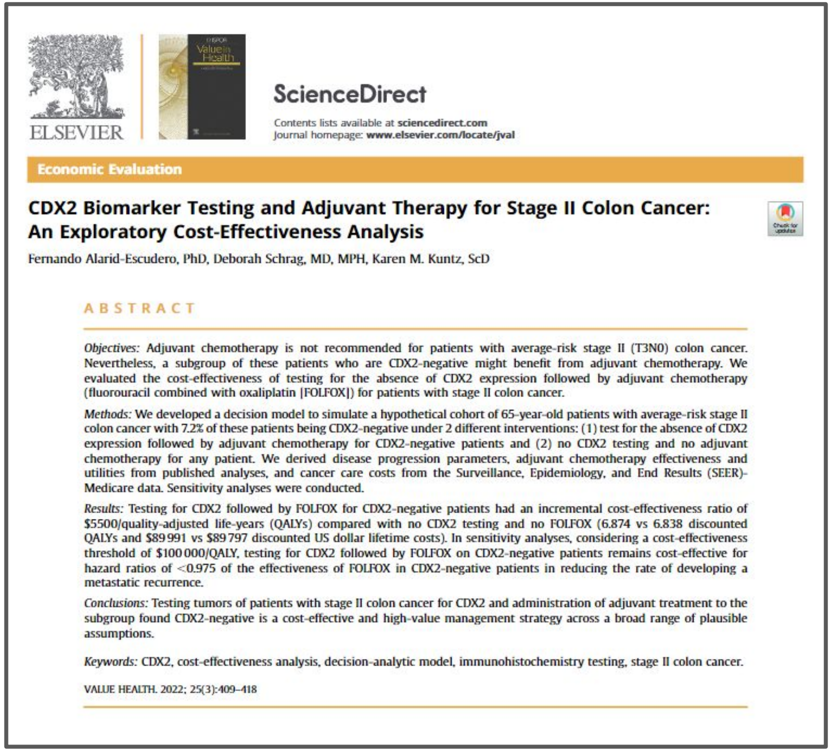
Benefits of using a Package vs non-packaged code.


Regulator preferred R Package
R for Health Technology Assessment
Model X Package

Benefits of using a Package vs non-packaged code.

- R Packages can serve as **templates for best-practice** in health economic model building.
- They make it **easier to review** since code is documented and unit tested by default.
- They make it **easier to distribute code** so that others can apply the same methods.
- **Confidence is crucial**, there is a key role for trusted experts to give legitimacy to packages.
- This role would be substantial and ongoing indefinitely … but …
- It would result in a huge **gain in efficiency and quality** of health economic models.
- For individual researchers, attribution would be beneficial for their profile.

Previous examples



ScienceDirect

Contents lists available at sciencedirect.com
Journal homepage: www.elsevier.com/locate/jval

Economic Evaluation

CDX2 Biomarker Testing and Adjuvant Therapy for Stage II Colon Cancer: An Exploratory Cost-Effectiveness Analysis

Fernando Alarid-Escudero, PhD, Deborah Schrag, MD, MPH, Karen M. Kuntz, ScD

ABSTRACT

Objectives: Adjuvant chemotherapy is not recommended for patients with average-risk stage II (T3N0) colon cancer. Nevertheless, a subgroup of these patients who are CDX2-negative might benefit from adjuvant chemotherapy. We evaluated the cost-effectiveness of testing for the absence of CDX2 expression followed by adjuvant chemotherapy (fluorouracil combined with oxaliplatin [FOLFOX]) for patients with stage II colon cancer.

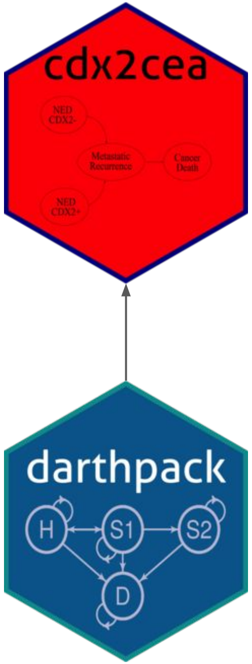
Methods: We developed a decision model to simulate a hypothetical cohort of 65-year-old patients with average-risk stage II colon cancer with 7.2% of these patients being CDX2-negative under 2 different interventions: (1) test for the absence of CDX2 expression followed by adjuvant chemotherapy for CDX2-negative patients and (2) no CDX2 testing and no adjuvant chemotherapy for any patient. We derived disease progression parameters, adjuvant chemotherapy effectiveness and utilities from published analyses, and cancer care costs from the Surveillance, Epidemiology, and End Results (SEER)-Medicare data. Sensitivity analyses were conducted.

Results: Testing for CDX2 followed by FOLFOX for CDX2-negative patients had an incremental cost-effectiveness ratio of $5500/quality-adjusted life-years (QALYs) compared with no CDX2 testing and no FOLFOX (6.874 vs 6.838 discounted QALYs and $89 991 vs $89 797 discounted US dollar lifetime costs). In sensitivity analyses, considering a cost-effectiveness threshold of $100 000/QALY, testing for CDX2 followed by FOLFOX on CDX2-negative patients remains cost-effective for hazard ratios of <0.975 of the effectiveness of FOLFOX in CDX2-negative patients in reducing the rate of developing a metastatic recurrence.

Conclusions: Testing tumors of patients with stage II colon cancer for CDX2 and administration of adjuvant treatment to the subgroup found CDX2-negative is a cost-effective and high-value management strategy across a broad range of plausible assumptions.

Keywords: CDX2, cost-effectiveness analysis, decision-analytic model, immunohistochemistry testing, stage II colon cancer.

VALUE HEALTH. 2022; 25(3):409–418



Alarid-Escudero, F., Schrag, D. and Kuntz, K.M., 2022. CDX2 biomarker testing and adjuvant therapy for stage II colon cancer: an exploratory cost-effectiveness analysis. *Value in Health*, *25*(3), pp.409-418.

R Packages for health economic evaluation: A tutorial.

Wellcome Open Research

Wellcome Open Research 2016 - DRAFT ARTICLE (PRE-SUBMISSION)

Packaging cost-effectiveness models in R: A tutorial.

Robert A. Smith*[1,2], Wael Mohammed[1,2], and Paul P. Schneider[1,2]

[1]School of Health and Related Research, University of Sheffield, Sheffield, S1 4DA, UK
[2]Dark Peak Analytics, Sheffield, S11 7BA, UK

Abstract

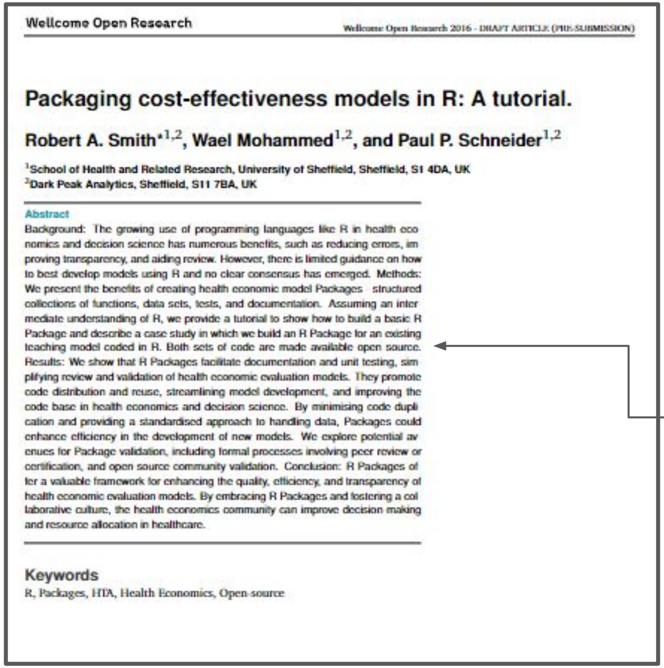
Background: The growing use of programming languages like R in health economics and decision science has numerous benefits, such as reducing errors, improving transparency, and aiding review. However, there is limited guidance on how to best develop models using R and no clear consensus has emerged. Methods: We present the benefits of creating health economic model Packages structured collections of functions, data sets, tests, and documentation. Assuming an intermediate understanding of R, we provide a tutorial to show how to build a basic R Package and describe a case study in which we build an R Package for an existing teaching model coded in R. Both sets of code are made available open source. Results: We show that R Packages facilitate documentation and unit testing, simplifying review and validation of health economic evaluation models. They promote code distribution and reuse, streamlining model development, and improving the code base in health economics and decision science. By minimising code duplication and providing a standardised approach to handling data, Packages could enhance efficiency in the development of new models. We explore potential avenues for Package validation, including formal processes involving peer review or certification, and open source community validation. Conclusion: R Packages offer a valuable framework for enhancing the quality, efficiency, and transparency of health economic evaluation models. By embracing R Packages and fostering a collaborative culture, the health economics community can improve decision making and resource allocation in healthcare.

Keywords

R, Packages, HTA, Health Economics, Open-source

Smith, R.A., Mohammed, W. and Schneider, P.P. (2023). Packaging cost-effectiveness models in R: A tutorial.
Draft paper currently under review in GoogleDoc

R Packages for health economic evaluation: A tutorial.



Wellcome Open Research

Wellcome Open Research 2016 - DRAFT ARTICLE (PRE-SUBMISSION)

Packaging cost-effectiveness models in R: A tutorial.

Robert A. Smith*[1,2], Wael Mohammed[1,2], and Paul P. Schneider[1,2]

[1]School of Health and Related Research, University of Sheffield, Sheffield, S1 4DA, UK
[2]Dark Peak Analytics, Sheffield, S11 7BA, UK

Abstract

Background: The growing use of programming languages like R in health economics and decision science has numerous benefits, such as reducing errors, improving transparency, and aiding review. However, there is limited guidance on how to best develop models using R and no clear consensus has emerged. Methods: We present the benefits of creating health economic model Packages structured collections of functions, data sets, tests, and documentation. Assuming an intermediate understanding of R, we provide a tutorial to show how to build a basic R Package and describe a case study in which we build an R Package for an existing teaching model coded in R. Both sets of code are made available open source. Results: We show that R Packages facilitate documentation and unit testing, simplifying review and validation of health economic evaluation models. They promote code distribution and reuse, streamlining model development, and improving the code base in health economics and decision science. By minimising code duplication and providing a standardised approach to handling data, Packages could enhance efficiency in the development of new models. We explore potential avenues for Package validation, including formal processes involving peer review or certification, and open source community validation. Conclusion: R Packages offer a valuable framework for enhancing the quality, efficiency, and transparency of health economic evaluation models. By embracing R Packages and fostering a collaborative culture, the health economics community can improve decision making and resource allocation in healthcare.

Keywords

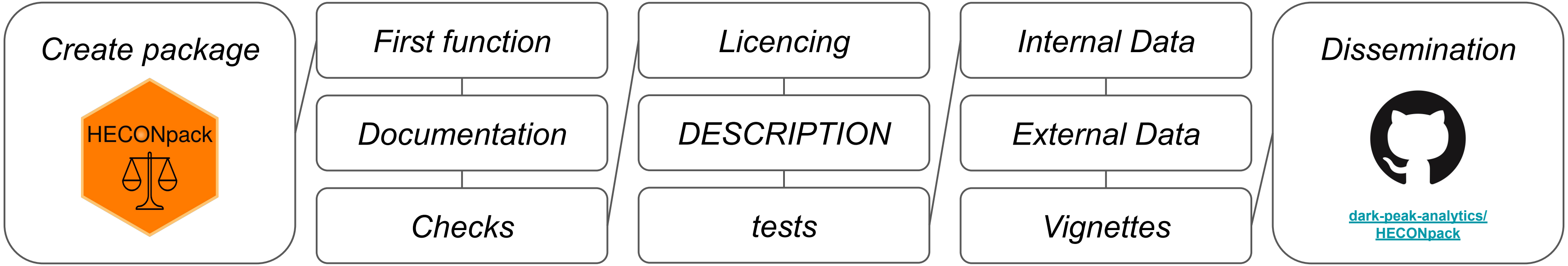
R, Packages, HTA, Health Economics, Open-source

This bit sucks … here's how to make it better!

(P.S. Make me a coauthor please)

Smith, R.A., Mohammed, W. and Schneider, P.P. (2023). Packaging cost-effectiveness models in R: A tutorial.
Draft paper currently under review in GoogleDoc

R Packages for health economic evaluation: A tutorial.



Create package

HECONpack

First function

Documentation

Checks

Licencing

DESCRIPTION

tests

Internal Data

External Data

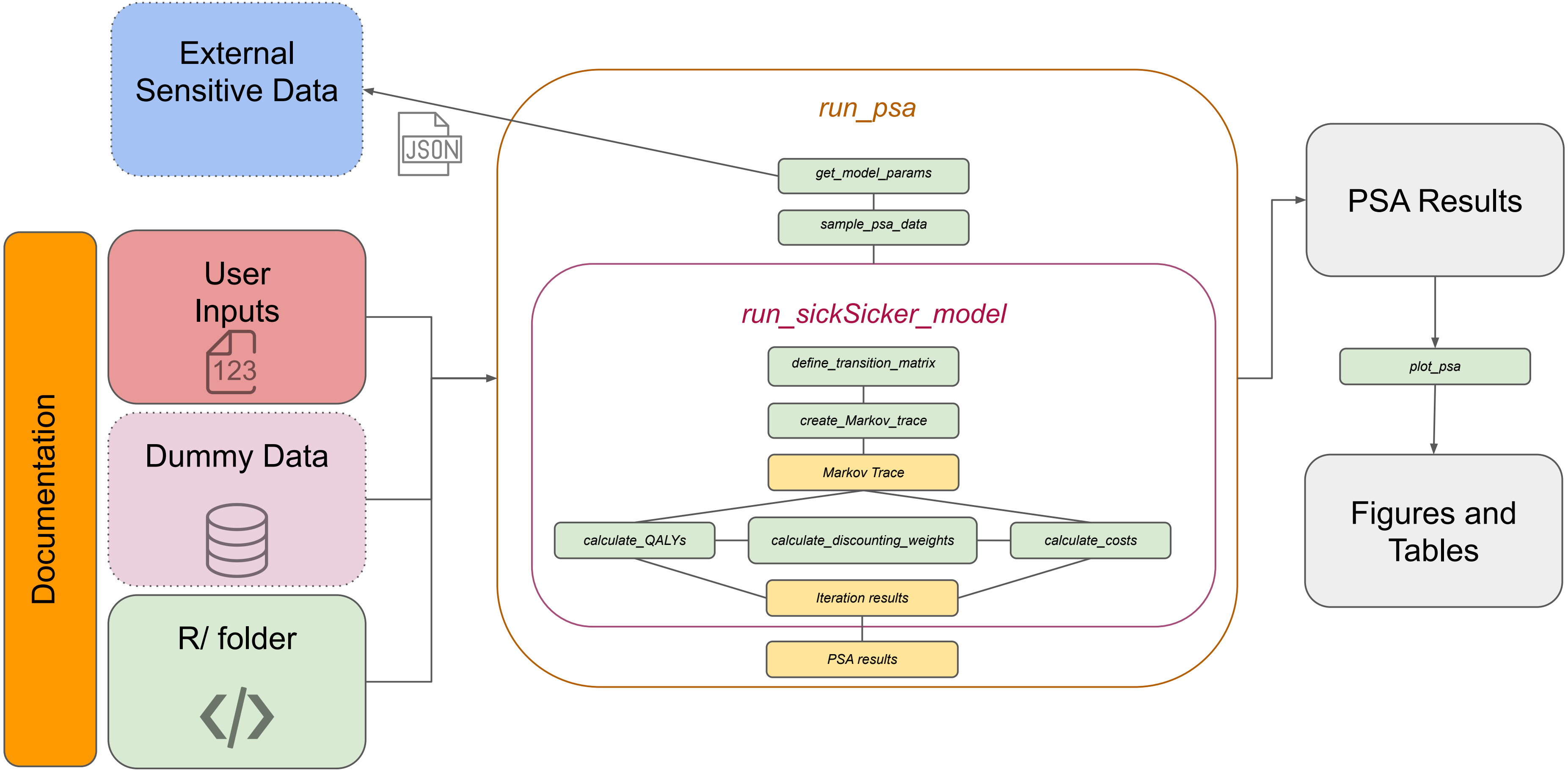
Vignettes

Dissemination

dark-peak-analytics/HECONpack

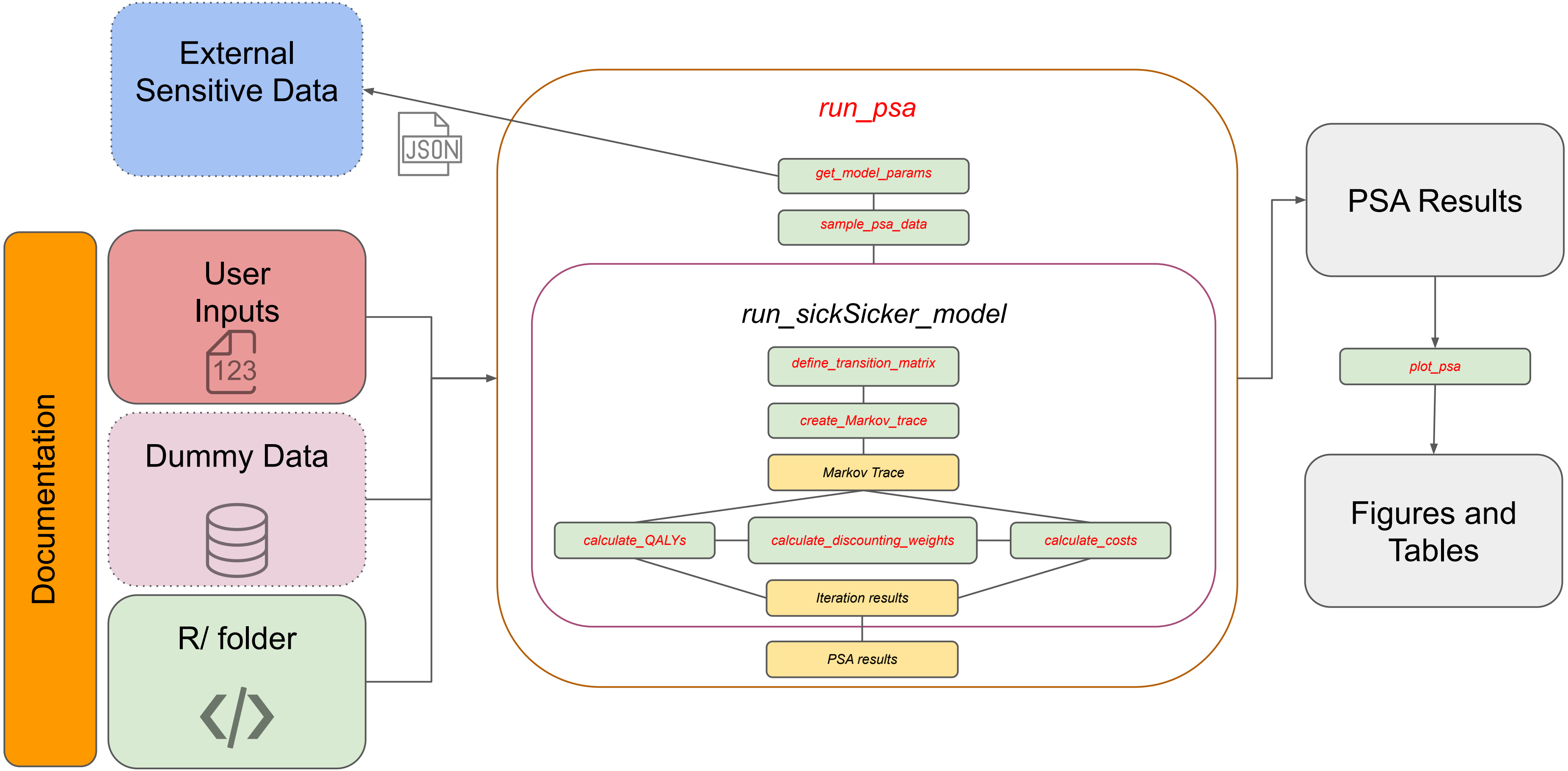
sicksickerPack

sicksickerPack process diagram



R demo

sicksickerPack process diagram



References

Alarid-Escudero, F., Schrag, D. and Kuntz, K.M., 2022. CDX2 biomarker testing and adjuvant therapy for stage II colon cancer: an exploratory cost-effectiveness analysis. *Value in Health*, *25*(3), pp.409-418.

Alarid-Escudero et al. An Introductory Tutorial on Cohort State-Transition Models in R Using a Cost-Effectiveness Analysis Example. Medical Decision Making, 2022

Smith, R.A., Mohammed, W. and Schneider, P.P. (2023). Packaging cost-effectiveness models in R: A tutorial.
Draft paper currently under review in GoogleDoc

Smith, R.A. (2023), *HECONpack* R package. https://github.com/dark-peak-analytics/HECONpack

Mohammed, W. & Smith, R.A. (2023). *sicksickerPack* R package https://github.com/dark-peak-analytics/sicksickerPack

Smith RA and Schneider PP. Making health economic models Shiny: A tutorial. Wellcome Open Res 2020, 5:69.

Smith RA, Schneider PP and **Mohammed W.** Living HTA: Automating Health Economic Evaluation with R Wellcome Open Res 2022, 7:194

– Thanks from Sheffield –

darkpeakanalytics.com/
contact@darkpeakanalytics.com/
github.com/dark-peak-analytics

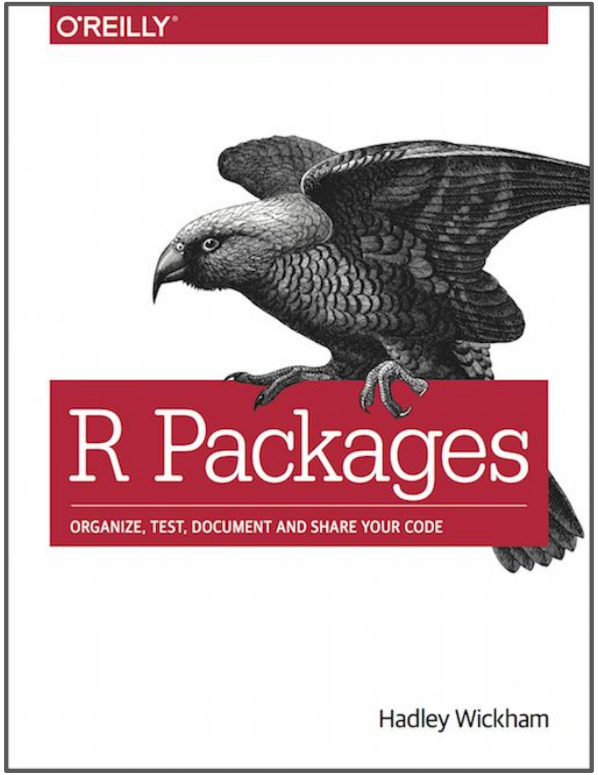
DARK PEAK ANALYTICS

University of Sheffield

APPENDIX
SHORT PACKAGES TUTORIAL

Packages - the basics

path	type
.Rbuildignore	file
DESCRIPTION	file
NAMESPACE	file
inst/	directory
man/	directory
R/	directory
data/	directory
vignettes/	directory
tests/	directory



https://r-pkgs.org/

Packages - the basics

path	type	description
.Rbuildignore	file	files to ignore when building package
DESCRIPTION	file	metadata, e.g. name and version.
NAMESPACE	file	from Roxygen, ensures names dependencies etc.
inst/	directory	installed files when user installs package
man/	directory	md files documenting for functions
R/	directory	R functions
data/	directory	data available within package
vignettes/	directory	generally used to showcase package functionality
tests/	directory	unit tests designed to ensure code works as intended

Building our first package

Step 1: Load devtools and friends

Step 2: Create your package skeleton

Step 3: Write first function

Step 4: Load & run function

Step 5: Run checks

Step 6: Write tests

Step 7: Repeat

https://r-pkgs.org/whole-game.html

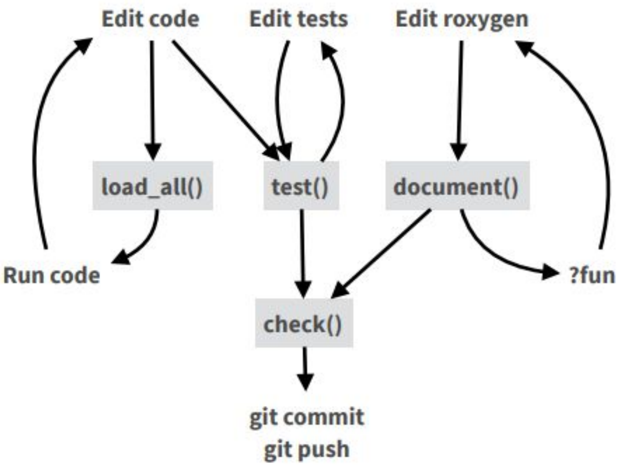
HECONpack

We are going to run through the process of building a highly innovative package which calculates an ICER from costs and QALYs for a baseline strategy and an intervention.

An existing teaching package can be found here:

[*https://github.com/dark-peak-analytics/HECONpack*](https://github.com/dark-peak-analytics/HECONpack)

HECONpack



- **load_all()** (Ctrl/Cmd + Shift + L) — Load code
- **document()** (Ctrl/Cmd + Shift + D) — Rebuild docs and NAMESPACE
- **test()** (Ctrl/Cmd + Shift + T) — Run tests
- **check()** (Ctrl/Cmd + Shift + E) — Check complete package

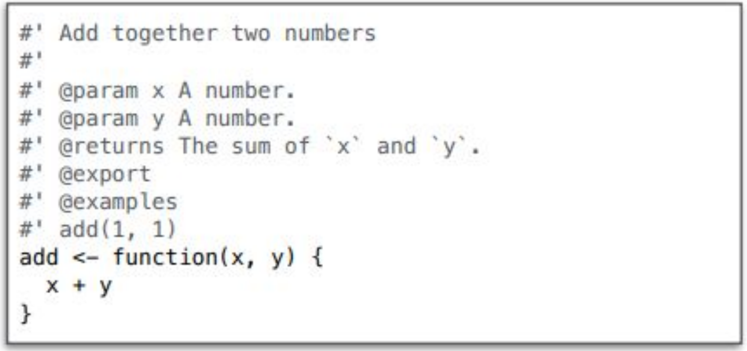
HECONpack

```
calcICER <-  function(e_int, e_base, c_int, c_base) {

  # calculate the incremental costs and effects
  inc_e <-  e_int - e_base
  inc_c <-  c_int - c_base

  # calculate the ICER
  icer <- inc_c / inc_e
  return(icer)
}
```

HECONpack



```
#' Add together two numbers
#'
#' @param x A number.
#' @param y A number.
#' @returns The sum of `x` and `y`.
#' @export
#' @examples
#' add(1, 1)
add <- function(x, y) {
  x + y
}
```

COMMON ROXYGEN TAGS

@description
@examples
@examplesIf
@export
@family
@inheritParams
@param
@rdname
@returns
@seealso

HECONpack

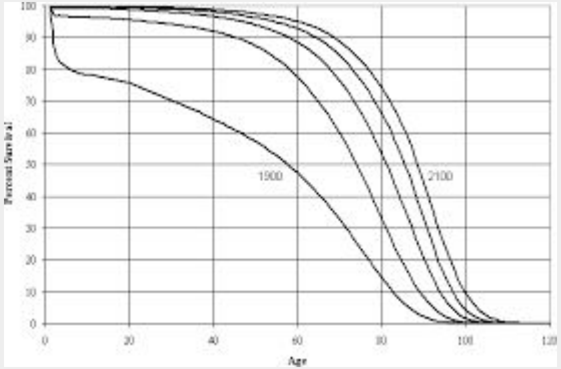
```
#' Calculate the Incremental Cost Effectiveness Ratio

#' Calculates the incremental effect and incremental costs of an intervention
#' compared to baseline and then uses the results to calculate the ICER

#' @param e_int  a single numeric value representing the effect (e.g. Total QALYs) in the intervention group.
#' @param e_base a single numeric value representing the effect (e.g. Total QALYs) in the base group.
#' @param c_int  a single numeric value representing the cost (e.g. Total £) in the intervention group.
#' @param c_base a single numeric value representing the cost (e.g. Total £) in the base group.
#' @return an single numeric value for the ICER.
#' @export
#' @examples
#' calcICER(e_int = 28.3, e_base = 22.5, c_int = 10000, c_base = 9200)
```

HECONpack

Including data



HECONpack

```
# INSTALLING PACKAGES ----
# we can now allow others to use functions from our package by
# installing the package from GitHub.
# NOTE: Up to now we have used source(<path to code>) but this is cumbersome.
remove.packages(pkgs = "HECONpack")
devtools::install_github("dark-peakanalytics/HECONpack")
```

HECONpack

Exercise

We are going to create a new package called ‘LTpack’ that contains a dataframe with life tables and a function to retrieve mortality rate a vector of age and sex.

1) Create a package skeleton with *devtools::create_package* function.
2) Add a function *add_nums* which adds two numbers together using *usethis::use_r* function.
3) Load the function using *devtools::load_all* function.
4) Add Roxygen documentation then document using *devtools::document*
5) Run checks using *devtools::check* and fix any issues (for example documentation, licencing, example fail).
6) Add a test using *use_test*
7) Read in the lifetable data from https://github.com/dark-peak-analytics/teaching_data/blob/main/ons_lifetables.csv and then create a dataset with *usethis::use_data* function.

Extensions

1) Write a function that retrieves age and sex specific mortality (either mx or qx) rates for a population.
2) Write a function that calculates mean annual mortality rates given the % male and female at a given age.

In both cases, don’t forget to test the function using *testthat* and to build in internal checks (e.g. with *assertthat*)